

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using SoundCapture;
using SoundAnalysis;

namespace FftGuitarTuner
{
    public class SoundFrequencyInfoSource : FrequencyInfoSource
    {
        Device device;
        Adapter adapter;
        int NotifyPointsInSecond;

        public SoundFrequencyInfoSource(Device device, int NotifyPointsInSecond)
        {
            this.device = device;
            this.NotifyPointsInSecond = NotifyPointsInSecond;
        }

        public override void Listen()
        {
            adapter = new Adapter(this, device, NotifyPointsInSecond);
            adapter.Start();
        }

        public override void Stop()
        {
            adapter.Stop();
        }

        // Sound Data Processing
        class Adapter : SoundCaptureBase // Descend from the soundcapture base
        {
            SoundFrequencyInfoSource owner;

            const double MinFreq = 60; // Open Low E: 82.4069Hz
            const double MaxFreq = 1300; // E6: 1319Hz

            public Adapter(SoundFrequencyInfoSource owner, Device device, int NotifyPointsInSecond)
                : base(device, NotifyPointsInSecond)
            {
                this.owner = owner;
            }

            // Process the sound capture data
            // Compute the spectrogram of the sound data using FFT
            // Find the maximum value(fundamental) in the spectrum
            // compute the fundamental frequency
            protected override void ProcessData(short[] data)
            {
                double[] x = new double[data.Length];
                for (int i = 0; i < x.Length; i++)
                {
                    x[i] = data[i];
                }

                // apply the fft to the captured sound data
                // This returns the frequency spectrum of the original complex signal
                double[] spectr = FftAlgorithm.Calculate(x);

                int index = 0;
                double max = spectr[0];

                // Compute the usefull range of data in the spectrum
                int usefullMaxSpectr = Math.Min(spectr.Length,
                    (int)(MaxFreq * spectr.Length / SampleRate) + 1);

                // find the maximum value (fundamental has the highest amplitude) in the spectrogram
                for (int i = 1; i < usefullMaxSpectr; i++)
                {
                    if (max < spectr[i])

```

```
        {
            max = spectr[i]; index = i;
        }

        // compute freq using the index of the maximum value found
        double freq = (double)SampleRate * index / spectr.Length;
        if (freq < MinFreq) freq = 0;

        // raise the frequency detected event
        owner.OnFrequencyDetected(new FrequencyDetectedEventArgs(freq));
    }
}
```